Efficient and Robust Physical 3DGS-MPM Simulation via Interior Filling and Text-Physics Optimization

Yikun Ma[1] **Yiqing Li**[1] **Jingwen Ye**[2] **Zhongkai Wu**[2] **Weidong Zhang**[2] **Lin Gao**[3]

Zhi Jin[1,4,5]*

[1]School of Intelligent Systems Engineering, Sun Yat-sen University, Shenzhen Campus, Shenzhen, Guangdong, China
[2]Independent Researcher
[3]Institute of Computing Technology, Chinese Academy of Sciences, China and University of Chinese Academy of Sciences, China, China
[4]Guangdong Provincial Key Laboratory of Fire Science and Intelligent Emergency Technology, Shenzhen, China
[5]Guangdong Provincial Key Laboratory of Robotics and Digital Intelligent Manufacturing Technology, Guangzhou, China

Abstract

Extending 3D Gaussian Splatting (3DGS) to 4D physical simulation remains challenging. Based on the Material Point Method (MPM), existing methods either rely on manual parameter tuning or distill dynamics from video diffusion models, limiting the generalization and optimization efficiency. Recent attempts using LLMs/VLMs suffer from a text/image-to-3D perceptual gap, yielding unstable physics behavior. In addition, they often ignore the surface structure of 3DGS, leading to implausible motion. We propose FastPhysGS, a fast and robust framework for physics-based dynamic 3DGS simulation: (1) Instance-aware Particle Filling (IPF) with Monte Carlo Importance Sampling (MCIS) to efficiently populate interior particles while preserving geometric fidelity; (2) Bidirectional Graph Decoupling Optimization (BGDO), an adaptive strategy that rapidly optimizes material parameters predicted from a VLM. Experiments show FastPhysGS achieves high-fidelity physical simulation in 1 minute using only 7 GB runtime memory, outperforming prior works with broad potential applications.

1 Introduction

Recently, remarkable progress in 3D reconstruction (1; 2; 3) and generation (4; 5; 6; 7) has significantly advanced applications in gaming, virtual reality, and robotics. However, generating realistic, dynamically consistent, and spatio-temporally coherent 4D contents remain challenging. With the progress in video generation models (8; 9; 10), several studies have explored constructing 4D content that reflects real-world dynamics (11; 12; 13). These methods typically leverage dynamic priors derived from video diffusion models to predict 4D deformations and subsequently reconstruct spatio-temporal scenes. Nevertheless, they inherently lack explicit physical constraints, often resulting in motions that violate physical laws and exhibit spatio-temporal inconsistencies of 4D content.

To address these issues, PhysGaussian (14) first integrates the Material Point Method (MPM) (15) with 3D Gaussian Splatting (3DGS) (2), leveraging an explicit Lagrangian-Eulerian grid representation

*Corresponding author. E-mail: jinzh26@mail.sysu.edu.cn

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

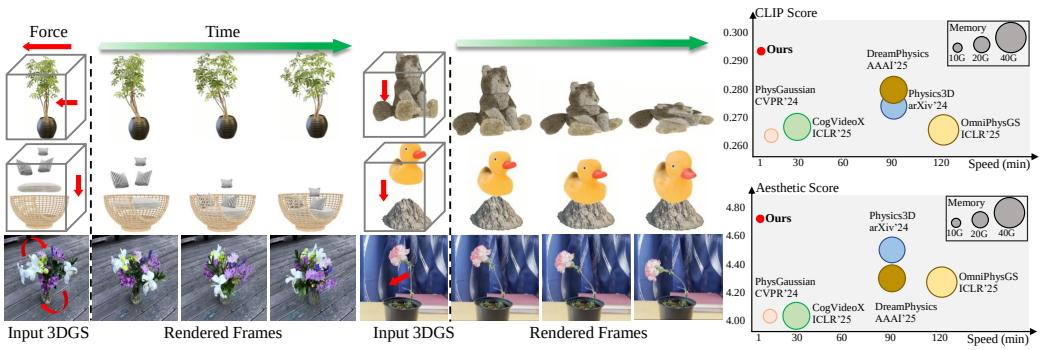


Figure 1: We propose FastPhysGS, an efficient and robust physics-based dynamic 3DGS simulation framework. Input a 3DGS scene, our method requires only 7 GB of memory and completes complex simulations within 1 minute, making it practical to generate real-time 4D physics-aware dynamics.

to simulate physically plausible dynamics in 3DGS. However, it requires manually specifying the physical material properties for each scene, which demands substantial expert knowledge. To mitigate this, subsequent methods (16; 17; 18) adopt Score Distillation Sampling (SDS) (4) to distill material properties from video diffusion models. However, this way is computationally expensive and relies on the motion priors from video diffusion models. In contrast, recent methods (19; 20; 21) employ LLMs/VLMs to predict physical parameters due to their reasoning capabilities. Nevertheless, a perceptual gap exists between text/images and native 3D content, which leads to unstable simulation results. Furthermore, existing methods overlook the hollow characteristic of 3DGS, resulting in unrealistic physical behaviors.

In summary, existing methods face three key challenges: **(1)** Neglecting the hollow 3DGS structure, while lacking the ability to distinguish instance-level filling. **(2)** Relying on manually designed parameters or diffusion distillation, leading to inadequate generalization and inefficient simulation. **(3)** Employing large models to predict physical parameters, leading to a perception gap with the native 3DGS content.

To address the aforementioned challenges, we propose FastPhysGS with two stages: **(1)** Considering the spatial structure of 3DGS, we propose Instance-aware Particle Filling (IPF). Our approach begins with segmenting the 3DGS objects, and leverages ray-casting guided by an occupancy field to estimate initial internal particles. To handle complex geometries filling, such as irregular surfaces and concave regions, we then design the Monte Carlo Importance Sampling (MCIS). **(2)** For adaptive and efficient refinement of physical parameters predicted by a VLM, we propose Bidirectional Graph Decoupling Optimization (BGDO). By exploring the stress-strain mechanics, BGDO leverages stress gradients and deformation to efficiently optimize parameters. Extensive experiments demonstrate that compared to other physical dynamic simulation methods, FastPhysGS generates plausible motions rapidly and robustly. As shown in Figures 1, our method achieves state-of-the-art results across a wide range of physical simulations and material types, only requires 7 GB running memory in 1 minute, which demonstrates outstanding practicality and broad potential application. Our main contributions are as follows:

(1) Real-Time, Memory-Efficient Dynamic 3DGS Simulation: FastPhysGS establishes a novel paradigm for physics-based dynamic 3DGS, achieving high-fidelity, multi-materials simulation with remarkable efficiency and physical plausibility.

(2) Instance-aware Complete Geometry Representation: We propose IPF with MCIS, which explicitly fills the hollow 3DGS, providing a geometrically complete and stable foundation for physical simulation.

(3) Perception-to-Physics optimization: We present BGDO, a novel adaptive refinement mechanism to rapidly correct VLM-predicted material parameters.

The rest of this paper is organized as follows: Sec. 2 reviews the related work. Sec. 3 introduces the preliminaries of 3DGS and MPM. Sec. 4 presents the details of our proposed method. Sec. 5 provides experimental analysis and results. Finally, Sec. 6 concludes the paper.

2 Related Work

2.1 4D Content Generation

4D content creation has grown significantly across various applications. With the success of video generation models (8; 9; 10), several methods propose to generate 4D content using video priors. MAV3D (22) first employs temporal Score Distillation Sampling (SDS) (4) from the text-to-video diffusion model. Animate124 (23) proposes an image-to-4D coarse-to-fine framework, while DreamGaussian4D (11) utilizes pre-generated videos to supervise the deformation of 3DGS. However, the diffusion dynamics may be physically inaccurate, and directly applying them leads to inplausible results.

2.2 Physics-Based Dynamic Simulation

To address the issues mentioned in Sec.2.1, recent works (14; 17; 24; 18; 16) introduce physics-based simulation. PhysGaussian (14) first coupled 3DGS (2) with continuum mechanics simulation via MPM (15). However, it requires manual configuration of physical parameters for each scene, causing inconvenient simulation. Subsequent approaches focus on automating MPM material perception. For instance, DreamPhysics (18) and Physics3D (16) leverage video generation models to estimate physical material parameters by SDS. However, it is time-consuming and relies on generative priors. To accelerate simulation, methods like PhysSplat (20) and Phys4DGen (19) utilize LLMs/VLMs to predict parameters. However, they rely on the output of large models, creating a perception gap with the actual 3D environment, leading to instable simulation. Moreover, existing works often overlook the hollow interior of 3DGS, which distorts the computation of particle stresses. Although PhysGaussian (14) introduces particle filling, its global processing makes it difficult to distinguish object instances and different materials.

3 Preliminaries

3.1 3D Gaussian Splatting

3DGS (2) represents a scene as a collection of 3D gaussians, each defined by a center $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$. The gaussian function at position $\boldsymbol{x}$ is given by:

$$G(\boldsymbol{x}) = \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right), \tag{1}$$

where the covariance matrix $\boldsymbol{\Sigma}$ is decomposed as $\boldsymbol{\Sigma} = \boldsymbol{RSS}^\top \boldsymbol{R}^\top$, where $\boldsymbol{S}$ is a diagonal scaling matrix and $\boldsymbol{R}$ is a rotation matrix. During rendering, view transformation $\boldsymbol{W}$ is applied, and the 2D projected covariance is computed using the Jacobian $\boldsymbol{J}$: $\boldsymbol{\Sigma'} = \boldsymbol{JW\Sigma W}^\top \boldsymbol{J}^\top$. The pixel color $\boldsymbol{Col}$ is computed by blending N overlapping gaussians:

$$\boldsymbol{Col} = \sum_{i=1}^{N} col_i \alpha_i \prod_{j<i}(1-\alpha_j), \tag{2}$$

where col_i and α_i denote the color and opacity of the i-th gaussian, respectively.

3.2 Material Point Method

MPM (15) is a widely used Lagrangian-Eulerian framework for physics-based simulation of continuum materials. In MPM, each time step consists of three sequential stages: (1) *Particle-to-Grid (P2G)*, where particle mass m and momentum are mapped to a background grid; (2) *Grid update*, where the discretized momentum equations are solved under applied internal and external forces $\mathbf{f}$; (3) *Grid-to-Particle (G2P)*, where updated velocities, positions, and deformation gradients are interpolated back to the particles. We adopt MLS-MPM (25), a MPM variant that improves momentum conservation via local affine velocity fields, and follow PhysGaussian to define each gaussian kernel as the time-dependent state:

$$\boldsymbol{x}_i(t) = \Delta(\boldsymbol{x}_i, t), \quad \boldsymbol{\Sigma_i(t)} = \boldsymbol{F}_i(t)\boldsymbol{\Sigma_i}\boldsymbol{F}_i(t)^\top, \tag{3}$$

where $\Delta(\cdot, t)$ and $\boldsymbol{F}_i(t)$ denote coordinate deformation and deformation gradient at timestep t.

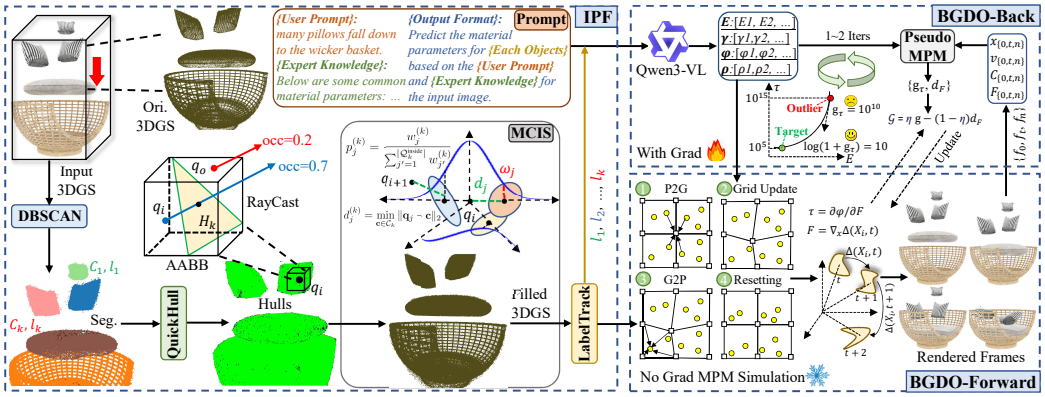


Figure 2: The pipeline of our method. The first stage IPF rapidly fills the interior 3DGS particles, while MCIS is designed to identify crucial points and handle complex geometries. The second stage BGDO is proposed to adaptively optimize MPM parameters. Overall, our method generates complete and realistic 4D physical dynamics in 1 minute, showcasing great potential in practical applications.

4 Method

The overall pipeline of FastPhysGS is illustrated in Figure 2, consisting of two stages: IPF and BGDO. Specifically, IPF populates the interior 3DGS particles, leveraging MCIS to handle complex geometry filling. Subsequently, BGDO employs the bidirectional graph decoupling strategy, which performs forward MPM simulation and backward optimization to refine physical parameters predicted from the VLM. Details of IPF and BGDO are described in Sec. 4.1 and 4.2.

4.1 Instance-aware Particle Filling

3DGS essentially reconstructs the surface appearance with empty internal structure. Therefore, it is prone to collapse during simulation, as the internal stress contribution is minimal. PhysGaussian (14) first utilizes the opacity field to determine internal points by checking whether a ray passes through grids with different opacities. However, this global filling makes it difficult to assign distinct materials to multiple objects (e.g., granting each object independent physical properties), while lacking reasonable adaptation for complex structures. For example, Figure 3 shows that PhysGaussian incorrectly fills the hollow region of the bamboo basket, causing the mat sides to be erroneously lifted upward.

To tackle these problems, we propose IPF to efficiently populate 3DGS particles with instance-aware capability. As shown in Figure 2, let the original 3DGS positions $\boldsymbol{\mathcal{P}}$ be:

$$\boldsymbol{\mathcal{P}} = \{\boldsymbol{p}_i \in \mathbb{R}^3\}_{i=1}^{N}, \tag{4}$$

where $\boldsymbol{p}_i$ denotes the mean of each gaussian, and N represents the total number of gaussians. Specifically, we apply DBSCAN clustering (26) to obtain K object instances for subsequent instance-aware MPM simulation:

$$\{\boldsymbol{\mathcal{C}_k}, l_k\}_{k=1}^{K} = \text{DBSCAN}(\boldsymbol{\mathcal{P}}, r), \boldsymbol{\mathcal{C}_k} = \{\boldsymbol{c}_j^{(k)} \in \mathbb{R}^3\}, \quad k = 1, 2, \ldots, K, \tag{5}$$

where $\boldsymbol{\mathcal{C}_k}$ denotes the k-th point cluster, l_k indicates the label of each cluster, r denotes the cluster radius, $\boldsymbol{c}_j$ represents the j-th point within the cluster $\boldsymbol{\mathcal{C}_k}$.

To rapidly obtain the interior gaussian points of each cluster $\boldsymbol{\mathcal{C}_k}$, we independently compute the convex hull utilizing Quickhull (27): $\boldsymbol{\mathcal{H}_k} = \text{Quickhull}(\boldsymbol{\mathcal{C}_k})$, where $\boldsymbol{\mathcal{H}_k}$ are the triangular meshes enclosing the points in $\boldsymbol{\mathcal{C}_k}$. For all possible candidate filling points $\boldsymbol{\mathcal{Q}_k}$ of each $\boldsymbol{\mathcal{C}_k}$, we construct AABB bounding boxes $\boldsymbol{\mathcal{B}_k}$ from $\boldsymbol{\mathcal{H}_k}$:

$$\boldsymbol{\mathcal{Q}_k} = \{\boldsymbol{q}_j \sim \text{Uni}(\boldsymbol{\mathcal{B}_k})\}_{j=1}^{M}, \quad M \gg |\mathcal{C}_k|, \quad \boldsymbol{\mathcal{B}_k} = [\boldsymbol{b}_{\min}^{(k)}, \boldsymbol{b}_{\max}^{(k)}] \subset \mathbb{R}^3, \tag{6}$$

where Uni represents uniform random sampling within the boxes, and $\boldsymbol{b}$ denotes the boundary value of the hull $\boldsymbol{\mathcal{H}_k}$.

To efficiently determine whether a candidate point $\boldsymbol{q}$ lies inside the convex hull $\boldsymbol{\mathcal{H}_k}$, we compute a 3D occupancy field $\text{occ}(\boldsymbol{q}; \boldsymbol{\mathcal{H}_k}) \in [0, 1]$, which employs fast ray-casting (28) to coarsely select the interior points based on occupancy probability threshold (experimentally set to 0.6):

$$\boldsymbol{\mathcal{Q}_k^{\text{inside}}} = \{\boldsymbol{q}_j \in \boldsymbol{\mathcal{Q}_k} \mid \text{occ}(\boldsymbol{q}_j; \boldsymbol{\mathcal{H}_k}) > 0.6\}\,. \tag{7}$$

However, relying solely on the occupancy field results in incorrect filling for irregularly shaped objects (e.g. Figure 7 illustrates the bamboo basket below, whose center is hollow and should not be filled). To address this, we propose the Monte Carlo Importance Sampling (MCIS) (29) strategy:

$$\mathbb{E}_p[f] = \int f(\boldsymbol{x})\, p(\boldsymbol{x})\, d\boldsymbol{x} = \int f(\boldsymbol{x})\, \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\, q(\boldsymbol{x})\, d\boldsymbol{x} \approx \frac{1}{n}\sum_{j=1}^{n} f(\boldsymbol{x}_j)\frac{p(\boldsymbol{x}_j)}{q(\boldsymbol{x}_j)} = \frac{1}{n}\sum_{j=1}^{n} f(\boldsymbol{x}_j) w_j\,, \tag{8}$$

this is a Monte Carlo expected value of any function $f(\boldsymbol{x})$ under distribution $p(\boldsymbol{x})$. The key insight lies in designing a non-uniform probability density function $p(x)$ to assign higher probabilities to important points from $\boldsymbol{\mathcal{Q}_k}$. We sample $\boldsymbol{x}_j \sim q(\boldsymbol{x})$, and provide the detailed importance sampling steps of MCIS. For each interior candidate point $\boldsymbol{q}_j \in \boldsymbol{\mathcal{Q}_k^{\text{inside}}}$, we compute its minimal Euclidean distance to the observed proxy $\boldsymbol{\mathcal{C}_k}$:

$$d_j^{(k)} = \min_{\boldsymbol{c} \in \boldsymbol{\mathcal{C}_k}} \|\boldsymbol{q}_j - \boldsymbol{c}\|_2. \tag{9}$$

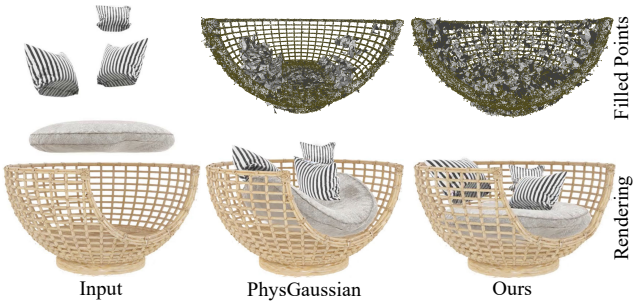


Figure 3: We extract the filled points as meshes for better visualization. PhysGaussian incorrectly fills the hollow region of the wicker basket, causing the mat to warp upward, while our method achieves accurate instance-aware filling.

Considering the distribution characteristic of 3DGS, we map distances into importance weights via an gaussian kernel, which exponentially attenuates influence with increasing distance. The unnormalized weights $w_j^{(k)}$ are converted into a discrete probability function over $\boldsymbol{\mathcal{Q}_k^{\text{inside}}}$:

$$w_j^{(k)} = \max\left(\exp\left(-\frac{(d_j^{(k)})^2}{2\sigma^2}\right), \epsilon\right),\ p_j^{(k)} = \frac{w_j^{(k)}}{\sum_{j'=1}^{|\boldsymbol{\mathcal{Q}_k^{\text{inside}}}|} w_{j'}^{(k)}},\ \epsilon = 10^{-6}, \tag{10}$$

where σ controls the locality of influence (empirically set to 0.02, see Sec. 5.3 for more details).

Then, we obtain n_k points via multinomial sampling according to $p_j^{(k)}$, yielding a refined subset. The final outputs are constructed by uniting the original $\boldsymbol{\mathcal{P}}$ with the sampled points:

$$\boldsymbol{\mathcal{P}}_{\text{filled}} = \bigcup_{k=1}^{K} \boldsymbol{\mathcal{S}_k},\quad \boldsymbol{\mathcal{P}}_{\text{all}} = \boldsymbol{\mathcal{P}} \cup \boldsymbol{\mathcal{P}}_{\text{filled}},\ \boldsymbol{\mathcal{S}_k} = \{\boldsymbol{s}_i^{(k)}\}_{i=1}^{n_k},\ \text{with } \mathbb{P}(\boldsymbol{s}_i^{(k)} = \boldsymbol{q}_j) = p_j^{(k)}, \tag{11}$$

where $\mathbb{P}$ represents the sampling probability, indicating $\boldsymbol{q}_j$ is sampled by an importance distribution. To prevent visual artifacts from $\boldsymbol{\mathcal{P}}_{\text{filled}}$, we nullify their appearance contribution by setting opacity $\alpha = 0$, retaining positional occupancy. Finally, to consistently retrieve dynamic gaussian parameters from a fixed label memory l_k, we propose label tracking as illustrated in Figure 4. In conclusion, IPF with MCIS fills the interior 3DGS particles, providing a geometrically complete structure foundation for subsequent MPM simulation.

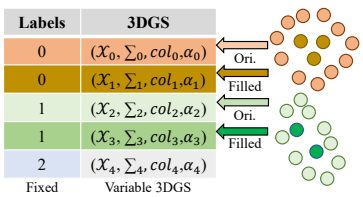


Figure 4: We use a contiguous memory to store 3DGS labels.

4.2 Bidirectional Graph Decoupling Optimization

Optimization Target Analysis. To conveniently obtain the physical parameters for MPM simulation, we employ Qwen3-VL (30) to obtain an initial prediction $\{\rho, \nu, \Phi, E\}$ for each

object instance l_k. Density ρ mainly influences the inertial term and has a relatively smooth effect. Poisson's ratio ν lies in a narrow range ($0 \sim 0.5$) and yields stable predictions thanks to rich pre-defined expert priors provided to the VLM. The energy density model Φ adopts a pre-defined form (e.g., Fixed Corotated Elasticity (31), StVK Elasticity (32), Drucker-Prager Plasticity (33), and Fluid Plasticity (34)), and we classify it by VLM. Thus, we optimize Young's modulus E, as it dominantly governs material stiffness. However, a perceptual gap between model reasoning and the native 3D space leads to unsuitable simulation, necessitating an optimization method to address this issue.

Specifically, we first derive the computational graph from E to the rendered frames $\boldsymbol{I}$:

$$\boldsymbol{I} \leftarrow \{\boldsymbol{\mathcal{X}}, \boldsymbol{\Sigma}, \alpha, col\} \leftarrow \{\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{C}, \boldsymbol{F}\} \leftarrow \boldsymbol{\tau} \leftarrow \Phi \leftarrow E, \tag{12}$$

where $\{\boldsymbol{\mathcal{X}}, \boldsymbol{\Sigma}, \alpha, col\}$ are the 3DGS parameters, and $\{\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{C}, \boldsymbol{F}\}$ represent the MPM particle states (position, velocity, affine momentum tensor, and deformation gradient), $\boldsymbol{\tau}$ is the first Piola–Kirchhoff stress. Since 3DGS parameters are directly generated from the MPM simulation, the rendered frames are merely visual representation of the underlying physical states. Therefore, we further analyze the numerical computations in the momentum exchange stage of MPM:

$$m_i^t \boldsymbol{v}_i^t = \sum_p N(\boldsymbol{x}_i - \boldsymbol{x}_p^t)\Big[m_p \boldsymbol{v}_p^t + \Big(m_p \boldsymbol{C}_p^t - \frac{4}{(\Delta x)^2}\Delta t V_p \frac{\partial \Phi}{\partial \boldsymbol{F}} \boldsymbol{F}_p^{tT}\Big)(\boldsymbol{x}_i - \boldsymbol{x}_p^t)\Big] + \boldsymbol{f}_i^t, \tag{13}$$

where $\partial\Phi/\partial\boldsymbol{F}$ denotes the particles stress $\boldsymbol{\tau}$, which captures the material-level forces in MPM. To conveniently control different material properties for multi-objects, we set the $\boldsymbol{\tau}$ as a variable term changing with different E in K lables:

$$\boldsymbol{\tau}(\boldsymbol{F}, E) = \sum_{k=1}^{K} \frac{\partial \Phi_k(\boldsymbol{F}_k, E_k)}{\partial \boldsymbol{F}_k}. \tag{14}$$

We aim to efficiently optimize E by leveraging $\{\boldsymbol{\tau}, \boldsymbol{F}\}$ as the physical plausibility signal, with two primary objectives: **(1) Time consumption:** avoid optimization during MPM simulation process, as current frame is determined by the previous frame, the full computation is highly time-consuming; **(2) Memory accumulation:** exclude simulation process from the computation graph, as it accumulates gradients across frames, leading to excessive GPU memory usage and out-of-memory errors.

Optimization Algorithm. Therefore, we propose BGDO to separate the forward simulation from backward optimization. As shown in Figure 2, we first perform a gradient-free forward MPM with a temporal and causal process, where subsequent frames are only influenced by prior states. Thus, we record only three key frames (initial frame $\boldsymbol{f_0}$, an intermediate frame $\boldsymbol{f_t}$, and the final frame $\boldsymbol{f_n}$) with attributes $\{\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{C}, \boldsymbol{F}\}$. This provides effective priors for optimization while minimizing storage costs. Then, we enable gradient computation in backward graph, and perform a **pseudo** simulation process, which preserves only the gradient pathways without actual simulation. To optimize E using $\boldsymbol{\tau}$ and $\boldsymbol{F}$, we leverage the stress gradient, while supervising material deformation by measuring the Frobenius norm (35) as guidance:

$$g_\tau = \frac{1}{3} \sum_{i \in \{0,t,n\}} \frac{\partial \|\boldsymbol{\tau_i}\|}{\partial E_i},\ d_{\boldsymbol{F}} = \frac{1}{3} \sum_{i \in \{0,t,n\}} \|\boldsymbol{F_i} - \mathbf{I}\|_F\,. \tag{15}$$

However, the stress gradient exhibits an wide dynamic range ($\sim 10^8$ to 10^{20}), causing numerical instability if used directly. Thus, we compress its magnitude via a logarithmic transform $g = \log(1 + g_\tau)$, and the optimization equation $\mathcal{G}$ is formulated as a weighted combination of g and $d_{\boldsymbol{F}}$:

$$\mathcal{G} = \eta \cdot g - (1 - \eta) \cdot d_{\mathrm{F}},\ \eta = min(1, 0.1 \cdot log(1 + E)), \tag{16}$$

where η is the guidance weight, which is proportional to E. Intuitively, an overly large E amplifies abnormal stress gradients, increasing the influence of the first term $\eta \cdot g$, while an excessively small E results in softer material and strengthens the effect of the deformation gradient in the second term $(1 - \eta) \cdot d_{\mathrm{F}}$. The balance between these two signals enables stable and adaptive material optimization. Finally, the update of Young's modulus E is defined as:

$$log(E) \leftarrow log(E) - \mathcal{G}\ . \tag{17}$$

Overall, BGDO is formulated from the perspective of numerical computation within the simulation, allowing for adaptive and rapid rectification of undesirable predicted parameters in only 1–2 iterations. IPF takes about **22s**, BGDO forward simulation takes **39s**, and the backward optimization occurs almost instantaneously, as shown in Table 1.

Table 1: The running time and memory across different modules and stages.

Method	DBSCAN	Occupancy	MCIS	MPM Forward	BGDO Backward	Full
Time	14 s	6 s	2 s	39 s	< 1 s	1 min
Memory	1.6 GB	1.8 GB	1.8GB	7 GB	7 GB	7 GB

5 Experiment

5.1 Implementation Details

Our method is deployed in the PyTorch (36) and Taichi (37) environments. We compare our method against four physics-based 3DGS simulation methods using the same dataset and hardware: PhysGaussian (14), DreamPhysics(18), Physics3D (16) and OmniPhysGS (17). Additionally, we also compare with SOTA video generation model CogVideoX-5B (9), and physics-aware large video model Veo-3.1 (38). All experiments are conducted on a single NVIDIA RTX A6000 GPU with 48 GB memory, with each method simulating and rendering 150 frames. We evaluate on three public datasets from (14; 16; 24) with 3DGS format. We additionally synthesize the multi-objects 3DGS dataset (e.g., a bear and a can, a duck and a stone). For more details, please refer to the Appendix.

5.2 Comparison with Baseline Methods

We conduct qualitative and quantitative comparisons, and compare the rendering results under identical camera trajectories and force priors against other methods.

Quantitative Comparison. For quantitative evaluation, we adopt the CLIP Score (CS) (39) to measure the semantic consistency between the frames and prompts, and the Aesthetic Score (AS) (40) to assess the visual fidelity. We further follow VideoPhy2 (41), and adopt Semantic Adherence (SA) to measure the video-text alignment fidelity, while Physical Commensense (PC) measures whether videos obey the physics laws of the real-world. We also compare the capability of automatic parameter optimization, running memory consumption, and simulation time. As shown in Table 2, compared to the CogVideoX, the physics-aware Veo achieves higher visual quality with AS 4.90 and physical realism with PC 0.49, yet it struggles to simulate dynamics that align with user intention. Regarding dynamic 3DGS simulation methods, although PhysGaussian achieves low simulation costs, it requires manual parameter tuning, and yields unsatisfactory text-image alignment and rendering quality with CS 0.267 and AS 4.06. In contrast, existing video distillation-based methods (Physics3D, DreamPhysics and OmniPhysGS) achieve higher scores in physical realism evaluation with SA 0.81 and PC 0.55, but incur high memory and computational cost.

By comparison, our FastPhysGS achieves the best performance with the lowest computational cost, requiring only 7 GB memory and 1 minute simulation time. These demonstrate potential values for interactive performance.

Qualitative Comparison. As shown in Figure 5, given the same prompt and 3DGS, along with identical motion patterns and force priors, we compare the rendering results under the same pose. We present results for single-object, multi-objects, and scene-level scenarios, respectively. CogVideoX struggles to produce physically plausible dynamics, while Veo generates high-fidelity results, it still differs from user intention. For example, we aim to generate a vertical tearing motion of the bread, CogVideoX and Veo fail to capture this interaction, resulting in inconsistent behaviors. Among physics-based 3DGS methods, OmniPhysGS lacks effective interior filling, leading to unrealistic deformation during collision (e.g., the bear hits a metal can with unnatural collapse). Other manual-

Table 2: Quantitative comparisons. We compare the CLIP Score (CS), Aesthetic Score (AS), Semantic Adherence (SA) and Physical Commensense (PC). The best results are highlighted in **bold**.

Method	Auto Param.	CS↑	AS↑	SA↑	PC↑	Memory (GB)↓	Time (min)↓
CogVideoX1.5-5B	-	0.269	4.04	0.44	0.21	33	24
Veo-3.1	-	0.291	**4.90**	0.81	0.49	-	-
PhysGaussian	✗	0.267	4.06	0.69	0.55	12	2
Physics3D	✔	0.275	4.48	0.81	0.53	30	90
DreamPhysics	✔	0.279	4.38	0.75	0.51	35	90
OmniPhysGS	✔	0.265	4.35	0.63	0.45	40	120
Ours	✔	**0.292**	4.71	**0.87**	**0.61**	**7**	**1**

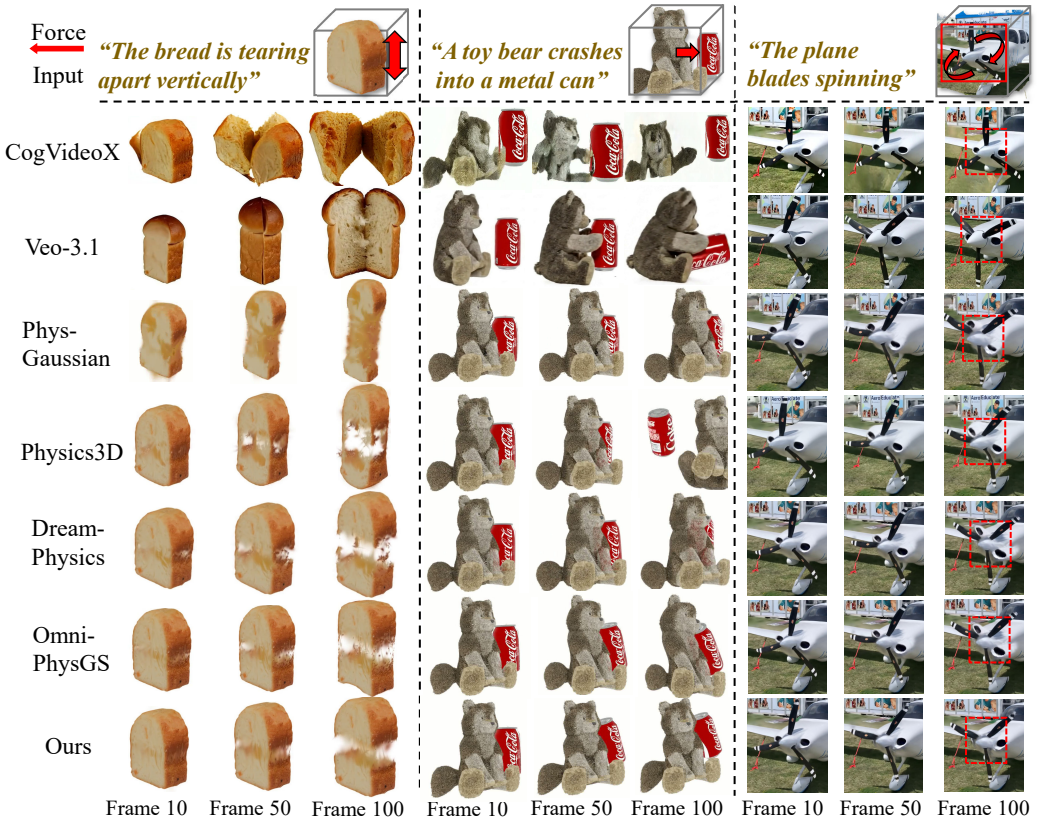


Figure 5: Visual comparisons between FastPhysGS and other physics-based 3DGS-MPM simulation and video generation methods. Our method achieves better dynamic visual performance while exhibiting the lowest simulation memory and the fastest execution speed.

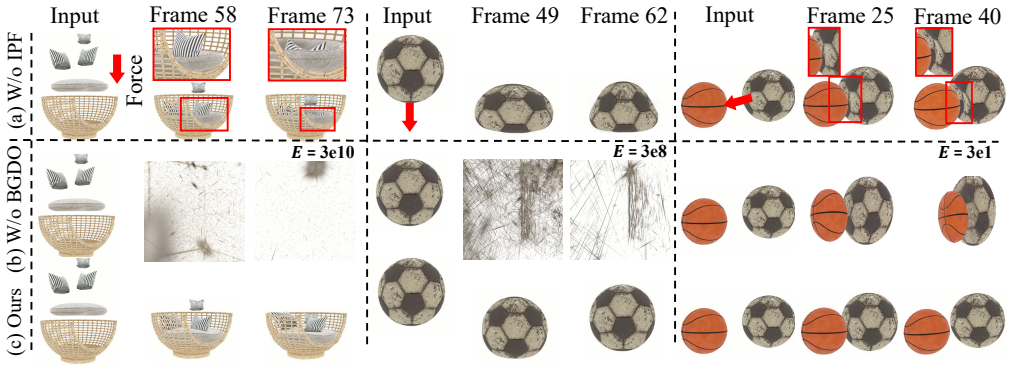


Figure 6: Visual comparisons of our ablation studies of FastPhysGS.

designed or distillation-based methods face challenges in achieving accurate instance-level simulation. For example, when simulating the rotation of aircraft blades, the static fuselage erroneously deforms. In comparison, our method benefits from IPF, providing a geometric foundation, while BGDO allows for correcting parameters from VLM with better stability.

5.3 Ablation Study

We conduct ablation experiments to evaluate the effectiveness of our proposed modules:

W/o IPF. As shown in Figure 6(a) and Table 3, missing particle filling causes internal collapse of 3DGS, leading to visually implausible results. By contrast, our IPF achieves interior filling, providing a geometrically complete 3D structure for stable and realistic MPM simulation.

Table 3: Quantitative metrics of our ablation study.

Method	CS↑	AS↑	SA↑	PC↑	Mem.	Time
w/o IPF	0.263	4.26	0.60	0.44	**5**	0.8
w/o MCIS	0.285	4.31	0.72	0.52	7	1
w/o BGDO	0.217	3.34	0.57	0.36	7	**0.5**
Ours	**0.292**	**4.71**	**0.87**	**0.61**	7	1



Figure 7: Visual ablation study of MCIS.

W/o MCIS. Figure 7(b) and Table 3 show that without MCIS, complex geometries like concave surfaces and irregular curved bodies are challenging to handle, resulting in inaccurate filling. MCIS mitigates this issue by modeling importance sampling, achieving correct completion.

σ of MCIS. We analyze the influence of different gaussian standard deviations σ within MCIS. Specifically, we observe the results for σ =1 (has no effect), 0.06, 0.04, and 0.02, as shown in Figure 7. Smaller σ assigns higher weights to points close to cluster centers and suppresses those farther away. As a result, sampling becomes highly focused on geometrically salient regions, such as boundaries and non-concave regions. In contrast, larger σ values flatten the weight distribution, making sampling nearly uniform and diminishing the role of importance.

W/o BGDO. To assess the robustness of BGDO under corrupted initial parameters, we set the initial Young's modulus to 3×10^{10}, 3×10^{8}, and 3×10^{1}. As shown in Figure 6(b) and Table 3, these settings lead to varying degrees of simulation failure. In contrast, our BGDO stabilizes the system, recovering a reasonable Young's modulus and thereby enabling robust 3DGS simulation.

Optimization stability of BGDO. To further validate the stability of BGDO, we evaluate the stress τ under different initial parameters. As illustrated in Figure 8, we change only the E while keeping other conditions identical, and plot over 60 sets of results. It shows that when E is small, the gradient effect weakens and deformation dominates, while a large E causes a sharp gradient increase. This phenomenon prevents the optimization from diverging into false directions, further enhancing the robustness of BGDO. For more ablation studies of DBSCAN clustering radius r and material compatibility, please refer to the appendix.

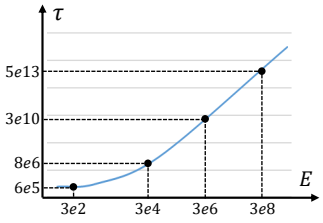


Figure 8: Variation of the stress τ with different E.

Relationship between Gaussians, cost, and performance. We conduct an ablation on the sampling ratio ζ (fraction of kept interior candidates), with results averaged across datasets.

Based on Table 4, we observe the following: Increasing the number of filled Gaussians initially improves quality metrics (CS, AS, SA, PC), but the performance saturates after $\zeta = 0.5$. Our default setting of $\zeta = 0.5$ achieves near-optimal performance while keeping memory and time consumption moderate. Further increasing the ratio to $\zeta = 0.8$ or 1.0 brings negligible gains in quality but significantly increases computational cost (e.g., time increases by 40% from $\zeta = 0.5$ to 1.0).

Physical Plausibility Validation. The video-based metrics evaluate physical plausibility only indirectly through the rendered appearance. Since obtaining real-world ground-truth dynamics for 3DGS is impractical, we further construct a controlled MPM benchmark to directly measure the physical accuracy of the simulated dynamics.

Table 4: Ablation study on the sampling ratio ζ.

Ratio ζ	Gaussians ($\times 10^3$)	Time (min)	Memory (GB)	CS ↑	AS ↑	SA ↑	PC ↑
0 (no IPF)	45	0.8	5.0	0.263	4.26	0.60	0.44
0.2	58 (+29%)	0.85	5.6	0.275	4.38	0.72	0.52
0.5	82 (+82%)	1.0	7.0	0.292	4.71	0.87	0.61
0.8	105 (+133%)	1.2	8.5	0.293	4.72	0.88	0.62
1.0 (all)	120 (+167%)	1.4	9.8	0.294	4.71	0.87	0.61

Table 5: Physical accuracy evaluation against the expert-reference MPM simulation.

Material	VLM-only		FastPhysGS (Ours)	
	$\mathcal{E}_{\text{traj}}\downarrow$	$\mathcal{E}_{\text{def}}\downarrow$	$\mathcal{E}_{\text{traj}}\downarrow$	$\mathcal{E}_{\text{def}}\downarrow$
Elastic	0.223	0.198	**0.038**	**0.031**
Plastic	0.267	0.241	**0.049**	**0.043**
Granular	0.302	0.276	**0.057**	**0.052**
Fluid	0.295	0.269	**0.054**	**0.048**
Average	0.272	0.246	**0.050**	**0.044**

For each material, experts specify the constitutive model and physical parameters $\theta^{\text{ref}} = \{\Phi^{\text{ref}}, E^{\text{ref}}, \rho^{\text{ref}}, \nu^{\text{ref}}\}$, which are used to generate the reference dynamics with MPM. We compare two settings against this reference: (1) VLM-only initialization, which directly simulates the material parameters estimated by the VLM, and (2) our full model with BGDO. The expert-reference, VLM-initialized, and FastPhysGS simulations share the same 3DGS particles, external forces, boundary conditions, and duration, so that the discrepancy originates solely from the estimated material parameters. Lower errors indicate that the simulated trajectories and deformations are closer to the known material response. We define the normalized trajectory error as:

$$\mathcal{E}_{\text{traj}} = \frac{1}{TND} \sum_{t=1}^{T} \sum_{i=1}^{N} \left\| \mathbf{x}_{i,t}^{\text{pred}} - \mathbf{x}_{i,t}^{\text{ref}} \right\|_2, \tag{18}$$

where N is the number of Gaussians, T is the number of frames, $\mathbf{x}_{i,t}$ is the position of the i-th Gaussian at frame t, and D is the diagonal length of the initial object bounding box. $\mathcal{E}_{\text{traj}}$ measures the complete motion response. Complementary to it, we define the normalized deformation error based on the Green-Lagrange strain tensor $\mathbf{S}_{i,t}$:

$$\mathbf{S}_{i,t} = \frac{1}{2}\left(\mathbf{F}_{i,t}^{\top}\mathbf{F}_{i,t} - \mathbf{I}\right), \quad \mathcal{E}_{\text{def}} = \frac{1}{TN} \sum_{t=1}^{T} \sum_{i=1}^{N} \frac{\left\| \mathbf{S}_{i,t}^{\text{pred}} - \mathbf{S}_{i,t}^{\text{ref}} \right\|_F}{\left\| \mathbf{S}_{i,t}^{\text{ref}} \right\|_F + \epsilon}, \tag{19}$$

where $\mathbf{F}_{i,t}$ is the deformation gradient and ϵ is a small constant for numerical stability. This metric removes rigid translation and rotation, and directly evaluates stretching, compression, and bending.

As shown in Table 5, FastPhysGS consistently reduces both errors across all four material types. On average, the trajectory error drops from 0.272 to 0.050, and the deformation error drops from 0.246 to 0.044. This indicates that BGDO not only improves simulation stability and rendered appearance, but also moves the simulated dynamics closer to the expert-reference physical response.

6 Conclusion and Limitations

We propose FastPhysGS for efficient and robust physical 3DGS simulation. IPF addresses instability caused by hollow 3DGS structures, while handling complex geometry filling with diverse instances. BGDO achieves rapid material parameters optimization. Experiments show the superior performance of FastPhysGS, enabling practical real-time simulation applications. However, FastPhysGS lacks the ability to generate new content. For example, removing a cup from a table would leave a visible hole. In future work, we will focus on integrating generative capabilities.

7 Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant 62671633, U24A20251 and Guangdong S&T Program No. 2024B1111060003.

References

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *European Conference on Computer Vision*. Springer, 2020, pp. 405–421.

[2] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.

[3] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny, "Vggt: Visual geometry grounded transformer," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 5294–5306.

[4] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," in *The Eleventh International Conference on Learning Representations*, 2023.

[5] Z. Wang, C. Lu, Y. Wang, F. Bao, C. Li, H. Su, and J. Zhu, "Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation," *Advances in neural information processing systems*, vol. 36, pp. 8406–8441, 2023.

[6] L. Höllein, A. Cao, A. Owens, J. Johnson, and M. Nießner, "Text2room: Extracting textured 3d meshes from 2d text-to-image models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 7909–7920.

[7] Y. Ma, D. Zhan, and Z. Jin, "Fastscene: text-driven fast 3d indoor scene generation via panoramic gaussian splatting," in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, 2024, pp. 1173–1181.

[8] T. Brooks, B. Peebles, C. Holmes, W. DePue, Y. Guo, L. Jing, D. Schnurr, J. Taylor, T. Luhman, E. Luhman *et al.*, "Video generation models as world simulators," *OpenAI Blog*, 2024.

[9] Z. Yang, J. Teng, W. Zheng, M. Ding, S. Huang, J. Xu, Y. Yang, W. Hong, X. Zhang, G. Feng *et al.*, "Cogvideox: Text-to-video diffusion models with an expert transformer," in *The Thirteenth International Conference on Learning Representations*, 2025.

[10] W. Kong, Q. Tian, Z. Zhang, R. Min, Z. Dai, J. Zhou, J. Xiong, X. Li, B. Wu, J. Zhang *et al.*, "Hunyuanvideo: A systematic framework for large video generative models," *arXiv preprint arXiv:2412.03603*, 2024.

[11] J. Ren, L. Pan, J. Tang, C. Zhang, A. Cao, G. Zeng, and Z. Liu, "Dreamgaussian4d: Generative 4d gaussian splatting," *arXiv preprint arXiv:2312.17142*, 2023.

[12] Y. Zeng, Y. Jiang, S. Zhu, Y. Lu, Y. Lin, H. Zhu, W. Hu, X. Cao, and Y. Yao, "Stag4d: Spatial-temporal anchored generative 4d gaussians," in *European Conference on Computer Vision*. Springer, 2024, pp. 163–179.

[13] J. Ren, C. Xie, A. Mirzaei, K. Kreis, Z. Liu, A. Torralba, S. Fidler, S. W. Kim, H. Ling *et al.*, "L4gm: Large 4d gaussian reconstruction model," *Advances in Neural Information Processing Systems*, vol. 37, pp. 56 828–56 858, 2024.

[14] T. Xie, Z. Zong, Y. Qiu, X. Li, Y. Feng, Y. Yang, and C. Jiang, "Physgaussian: Physics-integrated 3d gaussians for generative dynamics," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 4389–4398.

[15] A. Stomakhin, C. Schroeder, L. Chai, J. Teran, and A. Selle, "A material point method for snow simulation," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, pp. 1–10, 2013.

[16] F. Liu, H. Wang, S. Yao, S. Zhang, J. Zhou, and Y. Duan, "Physics3d: Learning physical properties of 3d gaussians via video diffusion," *NeurIPS*, 2024.

[17] Y. Lin, C. Lin, J. Xu, and Y. MU, "Omniphysgs: 3d constitutive gaussians for general physics-based dynamics generation," in *The Thirteenth International Conference on Learning Representations*, 2025.

[18] T. Huang, H. Zhang, Y. Zeng, Z. Zhang, H. Li, W. Zuo, and R. W. Lau, "Dreamphysics: Learning physics-based 3d dynamics with video diffusion priors," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 4, 2025, pp. 3733–3741.

[19] J. Lin, Z. Wang, D. Xu, S. Jiang, Y. Gong, and M. Jiang, "Phys4dgen: Physics-compliant 4d generation with multi-material composition perception," *arXiv preprint arXiv:2411.16800*, 2024.

[20] H. Zhao, H. Wang, X. Zhao, H. Fei, H. Wang, C. Long, and H. Zou, "Efficient physics simulation for 3d scenes via mllm-guided gaussian splatting," *ICCV*, 2025.

[21] B. Chen, H. Jiang, S. Liu, S. Gupta, Y. Li, H. Zhao, and S. Wang, "Physgen3d: Crafting a miniature interactive world from a single image," *CVPR*, 2025.

[22] U. Singer, S. Sheynin, A. Polyak, O. Ashual, I. Makarov, F. Kokkinos, N. Goyal, A. Vedaldi, D. Parikh, J. Johnson *et al.*, "Text-to-4d dynamic scene generation," in *Proceedings of the 40th International Conference on Machine Learning*, 2023, pp. 31 915–31 929.

[23] Y. Zhao, Z. Yan, E. Xie, L. Hong, Z. Li, and G. H. Lee, "Animate124: Animating one image to 4d dynamic scene," *arXiv preprint arXiv:2311.14603*, 2023.

[24] T. Zhang, H.-X. Yu, R. Wu, B. Y. Feng, C. Zheng, N. Snavely, J. Wu, and W. T. Freeman, "Physdreamer: Physics-based interaction with 3d objects via video generation," in *European Conference on Computer Vision*. Springer, 2024, pp. 388–406.

[25] Y. Hu, Y. Fang, Z. Ge, Z. Qu, Y. Zhu, A. Pradhana, and C. Jiang, "A moving least squares material point method with displacement discontinuity and two-way rigid body coupling," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[26] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[27] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software (TOMS)*, vol. 22, no. 4, pp. 469–483, 1996.

[28] M. Levoy, "Display of surface from volume data," *IEEE Computer Graphics and Applications*, vol. 8, no. 3, 1990.

[29] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, pp. 282–293.

[30] S. Bai, Y. Cai *et al.*, "Qwen3-vl technical report," *arXiv preprint arXiv:2511.21631*, 2025.

[31] A. Stomakhin, R. Howes, C. A. Schroeder, and J. M. Teran, "Energetically consistent invertible elasticity." in *Symposium on Computer Animation*, vol. 1, no. 2, 2012.

[32] J. Barbič and D. L. James, "Real-time subspace integration for st. venant-kirchhoff deformable models," *ACM transactions on graphics (TOG)*, vol. 24, no. 3, pp. 982–990, 2005.

[33] D. C. Drucker and W. Prager, "Soil mechanics and plastic analysis or limit design," *Quarterly of applied mathematics*, vol. 10, no. 2, pp. 157–165, 1952.

[34] A. Stomakhin, C. Schroeder, C. Jiang, L. Chai, J. Teran, and A. Selle, "Augmented mpm for phase-change and varied materials," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 1–11, 2014.

[35] C. F. Van Loan and G. Golub, "Matrix computations (johns hopkins studies in mathematical sciences)," *Matrix Computations*, vol. 5, p. 32, 1996.

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[37] Y. Hu, T.-M. Li, L. Anderson, J. Ragan-Kelley, and F. Durand, "Taichi: a language for high-performance computation on spatially sparse data structures," *ACM Trans. Graph.*, 2019.

[38] G. DeepMind, "Veo 3," 2025. [Online]. Available: https://deepmind.google/en/models/veo/

[39] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proceedings of the 38th International Conference on Machine Learning*, vol. 139, 2021, pp. 8748–8763.

[40] hristoph Schuhmann., "Laion-aesthetics," in *Online*, 2022.

[41] H. Bansal, C. Peng, Y. Bitton, R. Goldenberg, A. Grover, and K.-W. Chang, "Videophy-2: A challenging action-centric physical commonsense evaluation in video generation," *arXiv preprint arXiv:2503.06800*, 2025.

[42] J. Bonet and R. D. Wood, *Nonlinear continuum mechanics for finite element analysis*. Cambridge university press, 1997.

[43] G. Klár, T. Gast, A. Pradhana, C. Fu, C. Schroeder, C. Jiang, and J. Teran, "Drucker-prager elastoplasticity for sand animation," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–12, 2016.

A MPM Algorithm **14**

B Energy Constitutive Models **15**

C More Method Details of FastPhysGS **16**

C.1 Elasticity Models and Stress Sensitivity to E 16

C.2 Plasticity Models and Their Impact on Optimization 17

C.3 Deformation Guidance via Frobenius Norm 17

C.4 Gradient Compression and Log-Space Update 18

C.5 Algorithm and Limitation 18

D More Experimental Results **20**

D.1 More Setup Details 20

D.2 Ablation Study of DBSCAN 20

D.3 More Visualization Results 21

A MPM Algorithm

The Material Point Method (MPM) simulates the materials behavior by discretizing the continuum into particles and updating their properties in a hybrid Eulerian-Lagrangian way. MPM naturally handles large deformation and complex interactions with high robustness. The algorithm is summarized as follows:

Particle to Grid Transfer. In this step, mass m and momentum $m\mathbf{v}$ are transferred from particles to grid nodes by distributing the particle properties to nearby points. The accumulated mass and momentum at grid node i at time step n are computed as:

$$m_i^n = \sum_p w_{ip}^n m_p, \tag{20}$$

$$m_i^n \mathbf{v}_i^n = \sum_p w_{ip}^n m_p \left(\mathbf{v}_p^n + \mathbf{C}_p^n(\mathbf{x}_i - \mathbf{x}_p^n)\right), \tag{21}$$

where w_{ip}^n denotes the weighting function between particle p and grid point i, m_p is the particle mass, $\mathbf{v}_p^n$ is the particle velocity, $\mathbf{x}_p^n$ is the particle position, and $\mathbf{C}_p^n$ represents the local gradient correction term.

Grid Update. The grid velocities are updated based on external force and interaction with neighboring particles:

$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^n - \frac{\Delta t}{m_i} \sum_p \tau_p^n \nabla w_{ip}^n V_p^0 + \Delta t \mathbf{f}, \tag{22}$$

where Δt is the time step, τ_p^n is the stress tensor at particle p, V_p^0 is the reference volume, $\mathbf{f}$ is the external force, and m_i is the total mass accumulated at grid node i.

Grid to Particle Transfer. Velocities are interpolated back from the grid to the particles, updating their states:

$$\mathbf{v}_p^{n+1} = \sum_i \mathbf{v}_i^{n+1} w_{ip}^n, \tag{23}$$

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \mathbf{v}_p^{n+1}, \tag{24}$$

$$\mathbf{C}_p^{n+1} = \frac{4}{\Delta x^2} \sum_i w_{ip}^n \mathbf{v}_i^{n+1} (\mathbf{x}_i^n - \mathbf{x}_p^n)^T, \tag{25}$$

$$\nabla \mathbf{v}_p^{n+1} = \sum_i \mathbf{v}_i^{n+1} \nabla w_{ip}^{n\,T}, \tag{26}$$

$$\tau_p^{n+1} = \boldsymbol{\tau}(\mathbf{F}_\mathbf{E}^{n+1}, \mathbf{F}_\mathbf{N}^{n+1}), \tag{27}$$

where Δx is the Eulerian grid spacing, $\mathbf{F_E}$ and $\mathbf{F_N}$ represent the elastic and normal deformation gradients, respectively, and $\boldsymbol{\tau}$ is the constitutive stress model.

B Energy Constitutive Models

We integrate expert-designed constitutive models, which describe several representative materials including elasticity (e.g., rubber, branches, and cloth), plasticity (e.g., snow, metal, and clay), viscoelasticity (e.g., honey and mud), and fluidity (e.g., water, oil, and lava).

Fixed Corotated Elasticity

Following (31), we define fixed corotated elasticity as:

$$\mathbf{P}(\mathbf{F}) = 2\mu(\mathbf{F} - \mathbf{R}) + \lambda J(J-1)\mathbf{F}^{-T}, \tag{28}$$

where $\mathbf{R}$ is the rotation matrix obtained from the polar decomposition $\mathbf{F} = \mathbf{RS}$, $J = \det(\mathbf{F})$ is the volume change, and μ, λ are the Lamé parameters. This model is well-suited for simulating rubber-like materials due to its ability to capture large rotations with minimal shear distortion.

Neo-Hookean Elasticity

Based on (42), the Neo-Hookean elasticity is defined as:

$$\mathbf{P}(\mathbf{F}) = \mu \left(\mathbf{F} - \mathbf{F}^{-T}\right) + \lambda \log(J)\, \mathbf{F}^{-T}, \tag{29}$$

this formulation assumes isotropic hyperelastic behavior and is particularly effective in modeling spring-like elastic responses, such as those seen in soft tissues or idealized linear elastic solids.

StVK Elasticity

We adopt the St. Venant-Kirchhoff (StVK) model (32), which is given by:

$$\mathbf{P}(\mathbf{F}) = \mathbf{U}\left(2\mu\boldsymbol{\Sigma}^{-1}\ln\boldsymbol{\Sigma} + \lambda\,\mathrm{tr}(\ln\boldsymbol{\Sigma})\,\boldsymbol{\Sigma}^{-1}\right)\mathbf{V}^T, \tag{30}$$

where $\mathbf{F} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$ is the singular value decomposition (SVD) of the deformation gradient. The matrices $\mathbf{U}$, $\boldsymbol{\Sigma}$, and $\mathbf{V}$ represent the left stretch, singular values, and right rotation, respectively. The StVK model captures both elastic and plastic behaviors and is suitable for simulating materials such as sand and metals, especially when combined with plasticity models.

Identity Plasticity

The identity plasticity model assumes no plastic deformation, meaning the material behaves purely elastically. This is commonly used for idealized elastic materials:

$$\psi(\mathbf{F}) = \mathbf{F}, \tag{31}$$

this model serves as a baseline for comparison with more complex plasticity formulations.

Drucker-Prager Plasticity

Following (33; 43), we define Drucker-Prager plasticity using a return mapping based on the singular value decomposition (SVD) of the deformation gradient. Let $\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{\Sigma}$ contains the singular values. Define $\boldsymbol{\epsilon} = \log(\mathbf{\Sigma})$, which represents the logarithmic strain. The return mapping is then given by:

$$\psi(\mathbf{F}) = \mathbf{U}\mathcal{Z}(\mathbf{\Sigma})\mathbf{V}^T, \tag{32}$$

$$\mathcal{Z}(\mathbf{\Sigma}) = \begin{cases} \mathbf{I}, & \text{if } \sum(\boldsymbol{\epsilon}) > 0, \\ \mathbf{\Sigma}, & \text{if } \delta\gamma \leq 0 \text{ and } \sum(\boldsymbol{\epsilon}) \leq 0, \\ \exp\left(\boldsymbol{\epsilon} - \delta\gamma \frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|}\right), & \text{otherwise}, \end{cases} \tag{33}$$

where $\delta\gamma$ is the plastic update parameter, $\hat{\boldsymbol{\epsilon}}$ is the deviatoric part of $\boldsymbol{\epsilon}$, and $\sum(\boldsymbol{\epsilon})$ denotes the sum of the diagonal entries of $\boldsymbol{\epsilon}$. This model is particularly suitable for simulating granular materials such as snow and sand, due to its ability to capture dilatancy and yield under pressure.

von Mises Plasticity

Based on (25), von Mises plasticity is defined through a similar SVD-based return mapping:

$$\psi(\mathbf{F}) = \mathbf{U}\mathcal{Z}(\mathbf{\Sigma})\mathbf{V}^T, \tag{34}$$

$$\mathcal{Z}(\mathbf{\Sigma}) = \begin{cases} \mathbf{\Sigma}, & \text{if } \delta\gamma \leq 0, \\ \exp\left(\boldsymbol{\epsilon} - \delta\gamma \frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|}\right), & \text{otherwise}. \end{cases} \tag{35}$$

this formulation captures isotropic yielding and is widely used to simulate ductile materials such as metals and clay, where plastic flow occurs under shear stress.

Fluid Plasticity

We adopt the fluid plasticity model from (34), which treats the material as incompressible and fluid-like. The return mapping is defined as:

$$\psi(\mathbf{F}) = J^{1/3}\mathbf{I}, \tag{36}$$

where $J = \det(\mathbf{F})$ is the volume ratio. This model effectively removes all elastic deformation and enforces a uniform deformation state, making it suitable for simulating highly fluid-like materials such as water and lava.

C More Method Details of FastPhysGS

C.1 Elasticity Models and Stress Sensitivity to E

All elasticity models in our framework compute the first Piola-Kirchhoff stress $\boldsymbol{\tau} = \partial\Phi/\partial\mathbf{F}$ from the deformation gradient $\mathbf{F} \in \mathbb{R}^{3\times3}$, where Φ is the strain energy density.

Crucially, each model expresses $\boldsymbol{\tau}$ as a linear function of the Lamé parameters:

$$\mu = \frac{E}{2(1+\nu)}, \quad \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \tag{37}$$

which are themselves linear in Young's modulus E. This linearity implies that the stress scales approximately linearly with E, i.e., $\boldsymbol{\tau} \propto E$, making $\|\boldsymbol{\tau}\|$ a natural proxy for material stiffness.

We implement five models:

- **SigmaElasticity (Hencky/Logarithmic Strain)**: Uses principal logarithmic strains $\boldsymbol{\varepsilon} = \log(\mathbf{\Sigma})$, where $\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$. The stress is:

$$\boldsymbol{\tau} = \mathbf{U}\,\mathrm{diag}(2\mu\boldsymbol{\varepsilon} + \lambda\,\mathrm{tr}(\boldsymbol{\varepsilon})\mathbf{1})\mathbf{U}^\top. \tag{38}$$

  This model is energetically consistent and well-suited for large deformations.

- **CorotatedElasticity**: Removes rigid rotation via polar decomposition $\mathbf{R} = \mathbf{U}\mathbf{V}^\top$:

$$\boldsymbol{\tau} = 2\mu(\mathbf{F} - \mathbf{R})\mathbf{F}^\top + \lambda J(J-1)\mathbf{I}, \tag{39}$$

  where $J = \det(\mathbf{F})$. The deviatoric term captures shear, while the volumetric term enforces near-incompressibility.

- **StVKElasticity**: Based on Green–Lagrange strain $\mathbf{E} = \frac{1}{2}(\mathbf{F}^\top\mathbf{F} - \mathbf{I})$:

$$\boldsymbol{\tau} = 2\mu\mathbf{F}\mathbf{E} + \lambda J(J-1)\mathbf{I}. \tag{40}$$

  Accurate for moderate strains but may stiffen unrealistically under large compression.

- **FluidElasticity**: Sets $\mu = 0$, yielding purely volumetric response:

$$\boldsymbol{\tau} = \lambda J(J-1)\mathbf{I}. \tag{41}$$

  Suitable for liquids or highly dissipative materials.

- **VolumeElasticity**: Uses Mie–Grüneisen equation of state:

$$\boldsymbol{\tau} = \kappa\left(J - J^{-\gamma+1}\right)\mathbf{I}, \quad \kappa = \tfrac{2}{3}\mu + \lambda,\ \gamma = 2. \tag{42}$$

  Better stability under extreme compression.

In all cases, $\partial\boldsymbol{\tau}/\partial E$ exists and is non-zero, enabling gradient-based optimization of E . However, numerical instabilities—especially near $J \approx 0$ or during plastic yielding—cause $\|\nabla_E\|\boldsymbol{\tau}\|\|$ to span many orders of magnitude (10^8 – 10^{20}), necessitating robust normalization.

C.2 Plasticity Models and Their Impact on Optimization

Plasticity modifies $\mathbf{F}$ after the elastic stress computation, thereby indirectly affecting the stress-deformation relationship used in BGDO. We support four models:

- **IdentityPlasticity**: No modification; purely elastic.
- **SigmaPlasticity**: Enforces unilateral incompressibility by clamping $J = \det(\mathbf{F}) \in [0.05, 1.2]$ and resetting $\mathbf{F} = J^{1/3}\mathbf{I}$. This suppresses unrealistic expansion and stabilizes simulation, but reduces sensitivity to E in highly compressed regions.
- **VonMisesPlasticity**: Yields when deviatoric strain exceeds yield stress σ_y :

$$\text{if } \|\boldsymbol{\varepsilon}_{\text{dev}}\| > \frac{\sigma_y}{2\mu},\ \text{then project } \boldsymbol{\varepsilon} \text{ onto yield surface.} \tag{43}$$

  Since $\mu \propto E$, higher E raises the yield threshold, making the material appear more elastic—this nonlinearity is captured by BGDO's stress gradient signal.

- **DruckerPragerPlasticity**: Pressure-dependent yielding:

$$f(\boldsymbol{\tau}) = \|\boldsymbol{\tau}_{\text{dev}}\| + \alpha\,\mathrm{tr}(\boldsymbol{\tau}) - c \le 0, \tag{44}$$

  where α depends on friction angle ϕ , and c is cohesion. Compression increases resistance to yielding. Because $\mathrm{tr}(\boldsymbol{\tau}) \propto E$, the yield condition itself depends on E , creating a complex but differentiable coupling that BGDO exploits.

Critically, **all plasticity corrections are applied after stress computation**, so the gradient $\nabla_E\|\boldsymbol{\tau}\|$ remains well-defined and reflects pre-plastic elastic behavior—exactly the signal needed for stiffness calibration.

C.3 Deformation Guidance via Frobenius Norm

The Frobenius norm of the post-step deformation gradient deviation,

$$\delta = \|\mathbf{F}' - \mathbf{I}\|_F = \sqrt{\sum_{i=1}^{3}\sum_{j=1}^{3}\left(F'_{ij} - \delta_{ij}\right)^2}, \tag{45}$$

quantifies the magnitude of local strain accumulated over a single MPM substep, where $\mathbf{F}'$ is the updated deformation gradient after one explicit integration step and δ_{ij} is the Kronecker delta. Although this measure is not fully rotation-invariant (unlike invariants of $\mathbf{C} = \mathbf{F}^\top \mathbf{F}$), the small time step size Δt in our simulation ensures that rotational components in $\mathbf{F}'$ are close to identity, i.e., $\mathbf{F}' \approx \mathbf{I} + \nabla \mathbf{u}$ with displacement gradient $\nabla \mathbf{u}$ small. Consequently, $\|\mathbf{F}' - \mathbf{I}\|_F$ approximates the Euclidean norm of the infinitesimal strain tensor and serves as a practical proxy for visible deformation.

We set a target value $\delta_{\text{target}} = 0.1$, chosen empirically to balance visual responsiveness and physical plausibility:

- If $\delta \ll \delta_{\text{target}}$, the material exhibits negligible motion—indicating excessive stiffness—so we reduce the penalty on softness (effectively encouraging a decrease in E);
- If $\delta \gg \delta_{\text{target}}$, the material may sag, collapse, or exhibit numerical instability—suggesting insufficient stiffness—so we increase E .

This deformation-based signal resolves an inherent ambiguity in stress-only optimization: two materials with different E can produce similar stress magnitudes under static loading but exhibit drastically different dynamic responses. By incorporating δ , BGDO aligns parameter calibration with perceptual motion cues.

C.4 Gradient Compression and Log-Space Update

The raw gradient of the stress norm with respect to the log-scale modulus,

$$\mathbf{g}_\tau = \nabla_{\log E} \|\boldsymbol{\tau}\| = \frac{\partial \|\boldsymbol{\tau}\|}{\partial \log E}, \tag{46}$$

exhibits extreme dynamic range due to the nonlinear dependence of $\boldsymbol{\tau}$ on $\mathbf{F}$ and the exponential mapping $E = \exp(\log E)$. In regions near element inversion ($J = \det(\mathbf{F}) \to 0$) or plastic yield, $\|\boldsymbol{\tau}\|$ can change by orders of magnitude with tiny changes in E, leading to $\|\mathbf{g}_\tau\| \in [10^8, 10^{20}]$ in practice. Direct use of such gradients causes optimizer divergence or dominance by outlier particles.

To stabilize training, we apply a smooth, monotonically increasing compression function:

$$\tilde{g}_\tau = \log(1 + |\mathbf{g}_\tau|). \tag{47}$$

This transformation:

- Compresses the dynamic range;
- Preserves gradient sign and monotonicity—larger stress sensitivity still yields larger update magnitude;
- Reduces the influence of extreme outliers during spatial averaging across particles or temporal averaging across key frames.

The final update operates in log-space:

$$\log E \leftarrow \log E - \mathcal{G}, \quad \text{where } \mathcal{G} = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \left[\eta \cdot \tilde{g}_\tau^{(k)} - (1 - \eta) \cdot \delta^{(k)} \right], \tag{48}$$

with key frame set $\mathcal{K}$, and balancing weight $\eta \in [0, 1]$. Updating $\log E$ guarantees $E = \exp(\log E) > 0$ and induces multiplicative updates on E , which are more natural for scale parameters than additive ones. This formulation enables robust convergence within 1–2 iterations while using only three stored simulation snapshots, achieving minimal memory overhead.

C.5 Algorithm and Limitation

The BGDO optimization flow, summarized in Algorithm 1, decouples the causal forward simulation from the non-causal parameter update. By restricting gradient computation to three key frames and leveraging compressed stress gradients with Frobenius-norm deformation guidance, it achieves stable and memory-efficient material calibration.

Limitation For highly fine-grained complex scenes, interior filling may be limited by segmentation accuracy, thereby affecting material precision. Adopting a large segmentation model could alleviate this issue. In the future, we will focus on more general representations and physical interactions.

Table 6: List of Symbols and Their Meanings

Symbol	Type	Meaning
$\boldsymbol{\mu}$	Vector	Center of a 3D Gaussian
$\boldsymbol{\Sigma}$	Matrix	Covariance matrix of a 3D Gaussian
$\boldsymbol{\Sigma}'$	Matrix	Projected 2D covariance in screen space
α_i	Scalar	Opacity of the i -th Gaussian
N	Integer	Number of Gaussians
m	Scalar	Particle mass in MPM
$\mathbf{f}$	Vector	Force vector in MPM
$\Delta(\cdot, t)$	Function	Time-dependent spatial deformation map
$\boldsymbol{F}_i(t)$	Matrix	Deformation gradient of particle i at time t
t	Scalar	Simulation time step
$\boldsymbol{\mathcal{P}}$	Set	Original set of 3DGS particle positions
$\boldsymbol{p}_i$	Vector	Position of i -th original Gaussian ($= \boldsymbol{\mu}_i$)
K	Integer	Number of object instances after clustering
$\boldsymbol{\mathcal{C}_k}$	Set	Point cluster for instance k
l_k	Integer	Instance identifier for cluster k
r	Scalar	DBSCAN neighborhood radius
$\boldsymbol{c}_j^{(k)}$	Vector	j -th point in cluster $\boldsymbol{\mathcal{C}_k}$
$\boldsymbol{\mathcal{H}_k}$	Mesh	Convex hull of cluster k
$\boldsymbol{\mathcal{B}_k}$	AABB	Axis-aligned bounding box of $\boldsymbol{\mathcal{H}_k}$
$\boldsymbol{b}_{\min}^{(k)}, \boldsymbol{b}_{\max}^{(k)}$	Vectors	Min/max corners of AABB for instance k
$\boldsymbol{\mathcal{Q}_k}$	Set	Candidate filling points
M	Integer	Number of candidate points
$\mathrm{occ}(\boldsymbol{q}; \boldsymbol{\mathcal{H}_k})$	Scalar	Occupancy probability
$\boldsymbol{\mathcal{Q}}_{\boldsymbol{k}}^{\text{inside}}$	Set	Interior candidates
$d_j^{(k)}$	Scalar	Minimal Euclidean distance
σ	Scalar	Gaussian kernel scale
ϵ	Scalar	Small constant (10^{-6}) for numerical stability
$w_j^{(k)}$	Scalar	Importance weight
$p_j^{(k)}$	Scalar	Normalized sampling probability
n_k	Integer	Number of filled points selected for instance k
$\boldsymbol{\mathcal{S}_k}$	Set	Final sampled interior points for instance k
$\boldsymbol{\mathcal{P}}_{\text{filled}}$	Set	All filled points
$\boldsymbol{\mathcal{P}}_{\text{all}}$	Set	Full particle set
ρ	Scalar	Material density
ν	Scalar	Poisson's ratio
Φ	Function	Energy density model
E	Scalar	Young's modulus
$\boldsymbol{I}$	Image	Rendered frames
$\boldsymbol{\mathcal{X}}$	Set	3D positions of all Gaussians
$\boldsymbol{v}$	Vector	Particle velocity in MPM
$\boldsymbol{C}$	Matrix	Affine velocity field
$\boldsymbol{\tau}$	Tensor	First Piola–Kirchhoff stress
$\boldsymbol{f_0}, \boldsymbol{f_t}, \boldsymbol{f_n}$	Frames	Key frames (initial, mid, final)
g_τ	Scalar	Average stress gradient
$d_{\boldsymbol{F}}$	Scalar	Average Frobenius norm of $\boldsymbol{F} - \mathbf{I}$
g	Scalar	Log-compressed stress gradient
$\mathcal{G}$	Scalar	Combined optimization signal
η	Scalar	Adaptive weight

Algorithm 1 Bidirectional Graph Decoupling Optimization (BGDO)

Require: Initial Young's modulus $\log E$, material labels $\{l_k\}$, total frames T, key frames $\mathcal{K} = \{0, t, n\}$, target deformation magnitude δ_{target}, balancing weight $w \in [0, 1]$
Ensure: Optimized $\log E$

Stage 1: Gradient-Free Forward Simulation
Initialize particle states $(\mathbf{x}, \mathbf{v}, \mathbf{C}, \mathbf{F})$
`frame_buffer` $\leftarrow \emptyset$
for frame $= 0$ to $T - 1$ **do**
  if frame $\in \mathcal{K}$ **then**
    Store snapshot: `frame_buffer` $\leftarrow$ `frame_buffer` $\cup \{(\mathbf{x}, \mathbf{v}, \mathbf{C}, \mathbf{F})\}$
  end if
  for substep $= 1$ to S **do** ▷ S : MPM substeps per frame
    $\boldsymbol{\tau} \leftarrow \text{ELASTICITY}(\mathbf{F}, \log E.\text{detach}())$
    $(\mathbf{x}, \mathbf{v}, \mathbf{C}, \mathbf{F}) \leftarrow \text{MPMSTEP}(\mathbf{x}, \mathbf{v}, \mathbf{C}, \mathbf{F}, \boldsymbol{\tau})$
    $\mathbf{F} \leftarrow \text{PLASTICITY}(\mathbf{F}, \log E.\text{detach}())$
  end for
end for
Stage 2: Gradient-Based Backward Optimization
if `frame_buffer` $\neq \emptyset$ **then**
  $\mathbf{g}_{\text{total}} \leftarrow \mathbf{0}$
  for each $(\mathbf{x}_i, \mathbf{v}_i, \mathbf{C}_i, \mathbf{F}_i) \in$ `frame_buffer` **do**
    $\mathbf{F}_i \leftarrow \mathbf{F}_i.\text{requires_grad}()$
    $\boldsymbol{\tau}_i \leftarrow \text{ELASTICITY}(\mathbf{F}_i, \log E)$
    $(_, _, _, \mathbf{F}'_i) \leftarrow \text{MPMSTEP}(\mathbf{x}_i, \mathbf{v}_i, \mathbf{C}_i, \mathbf{F}_i, \boldsymbol{\tau}_i)$
    $\mathbf{F}'_i \leftarrow \text{PLASTICITY}(\mathbf{F}'_i, \log E)$
  // Stress gradient signal (prevent over-stiffness)
    $s_i \leftarrow \|\boldsymbol{\tau}_i\|_F$
    $\mathbf{g}_\tau \leftarrow \nabla_{\log E}\, s_i$
    $\tilde{g}_\tau \leftarrow \log(1 + |\mathbf{g}_\tau|)$
  // Deformation guidance (prevent over-softness)
    $\delta_i \leftarrow \|\mathbf{F}'_i - \mathbf{I}\|_F$
    $\tilde{g}_\delta \leftarrow \delta_i - \delta_{\text{target}}$
    $\mathbf{g}_{\text{total}} \leftarrow \mathbf{g}_{\text{total}} + w \cdot \tilde{g}_\tau - (1 - w) \cdot \tilde{g}_\delta$
  end for
  $\log E \leftarrow \log E - \frac{1}{|\mathcal{K}|} \mathbf{g}_{\text{total}}$
end if
return $\log E$

D More Experimental Results

D.1 More Setup Details

To synthesize the multi-objects dataset, we collect public single-object datasets and then modify the underlying 3DGS code to enable the fusion of single-object scenes. In MPM, spatial normalization is applied to a $1 \times 1 \times 1$ volume, and the object center is placed at (0.5, 0.5, 0.5). The simulation runs for 150 frames, the prompt is generated by a large language model, and all other parameters, such as camera poses and viewing angles are determined by the input 3DGS scene.

D.2 Ablation Study of DBSCAN

To more intuitively illustrate the impact of different radius values on DBSCAN clustering, we visualize the segmentation results obtained with radii ranging from 0.01 to 0.06 in Figure 9. As can be observed, a radius of 0.01 exhibits almost no discriminative capability, yielding very poor segmentation. As the radius increases, the segmentation quality gradually improves. Starting from a radius of 0.05, the segmentation becomes accurate, albeit at the cost of increased computational time. In practice, we typically select a radius in the range of 0.05 to 0.07.

D.3 More Visualization Results

We present more visualization results in Figures 10 to 15, showcasing a rich variety of material simulations, such as rubber, elastomers, sand, liquids, flexible materials, soft bodies, and rigid bodies. These further demonstrate the superiority and practicality of our FastPhysGS.

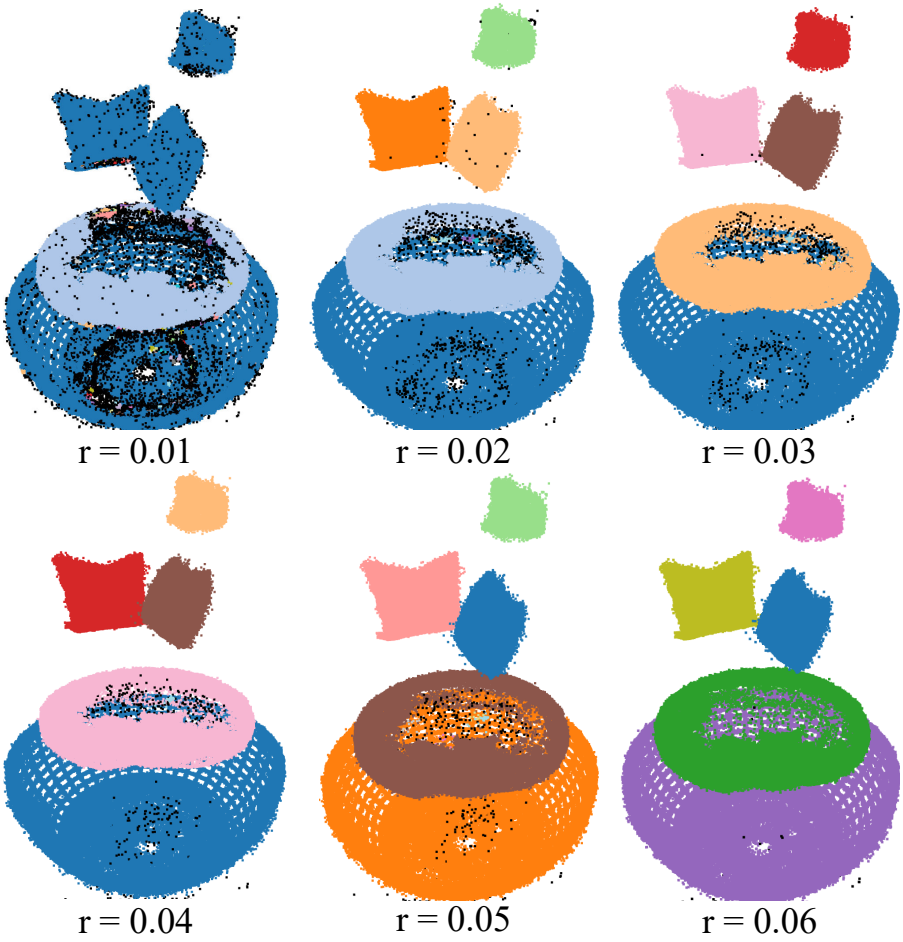
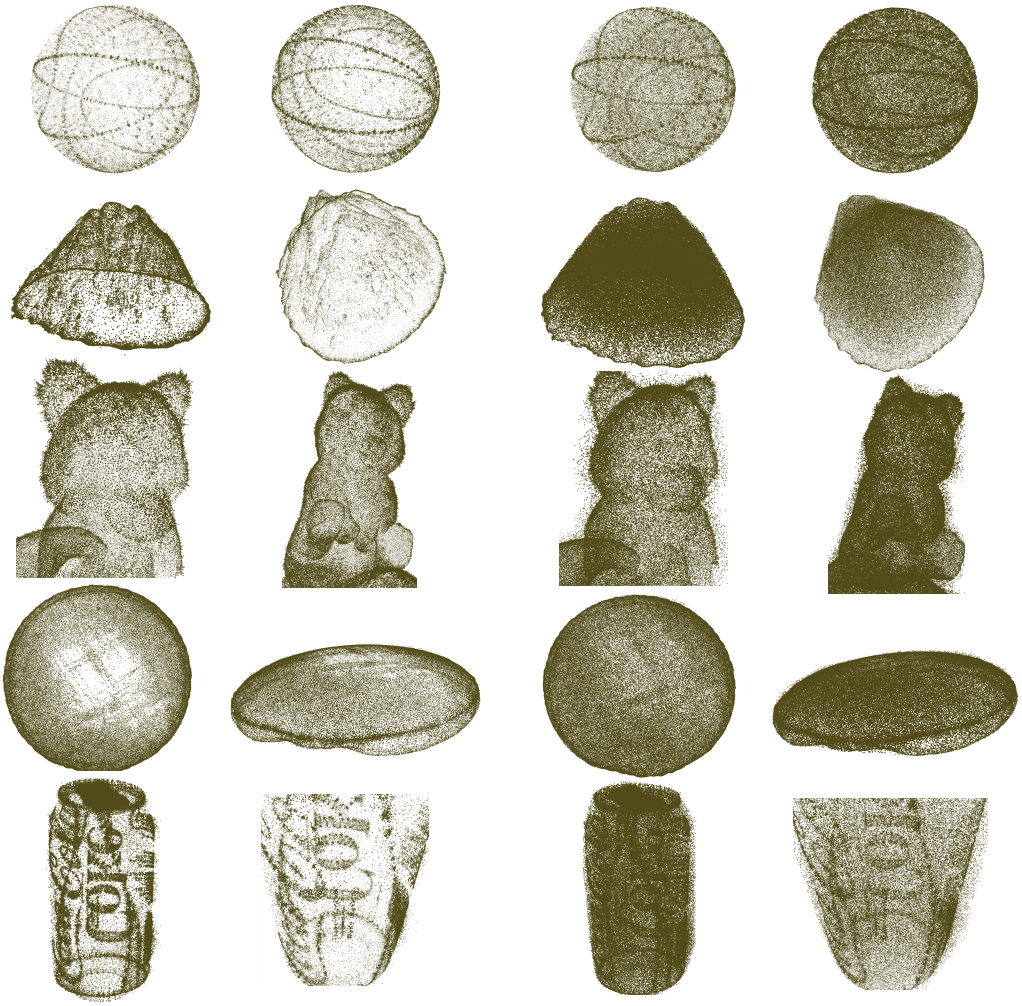


Figure 9: Ablation study of different r of DBSCAN.



Original 3DGS Filled 3DGS by our IPF

Figure 10: More visualized results of our IPF, demonstrating its ability to achieve stable filling across diverse structures.

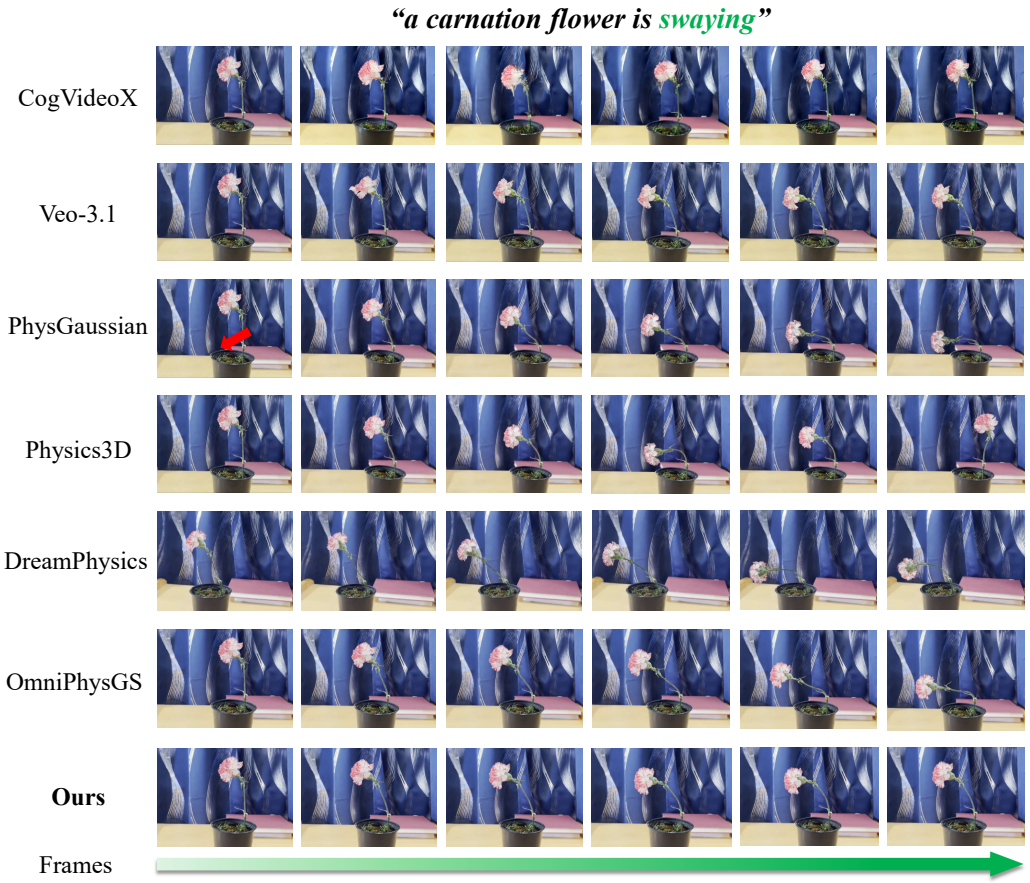


Figure 11: More visual comparisons of FastPhysGS and other baseline methods. Given the same prompt and force prior, our method produces more realistic dynamics. For example, this flower swings to a certain extent and then swings back.

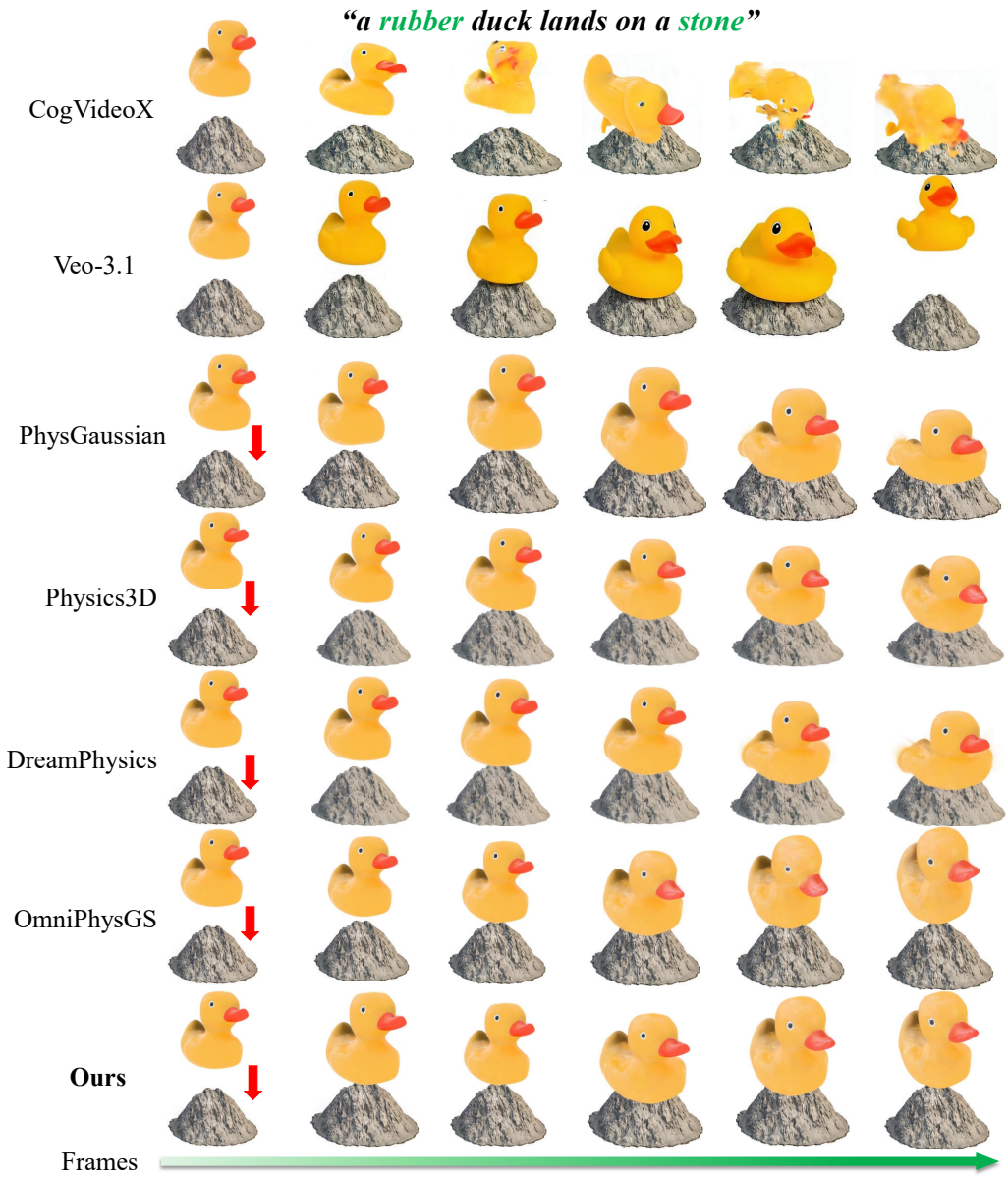


Figure 12: More visual comparisons of FastPhysGS and other baseline methods. Given the same text description and force priors, our method generates more realistic dynamic effects. For example, the rubber duck lands on a stone and bounces back.

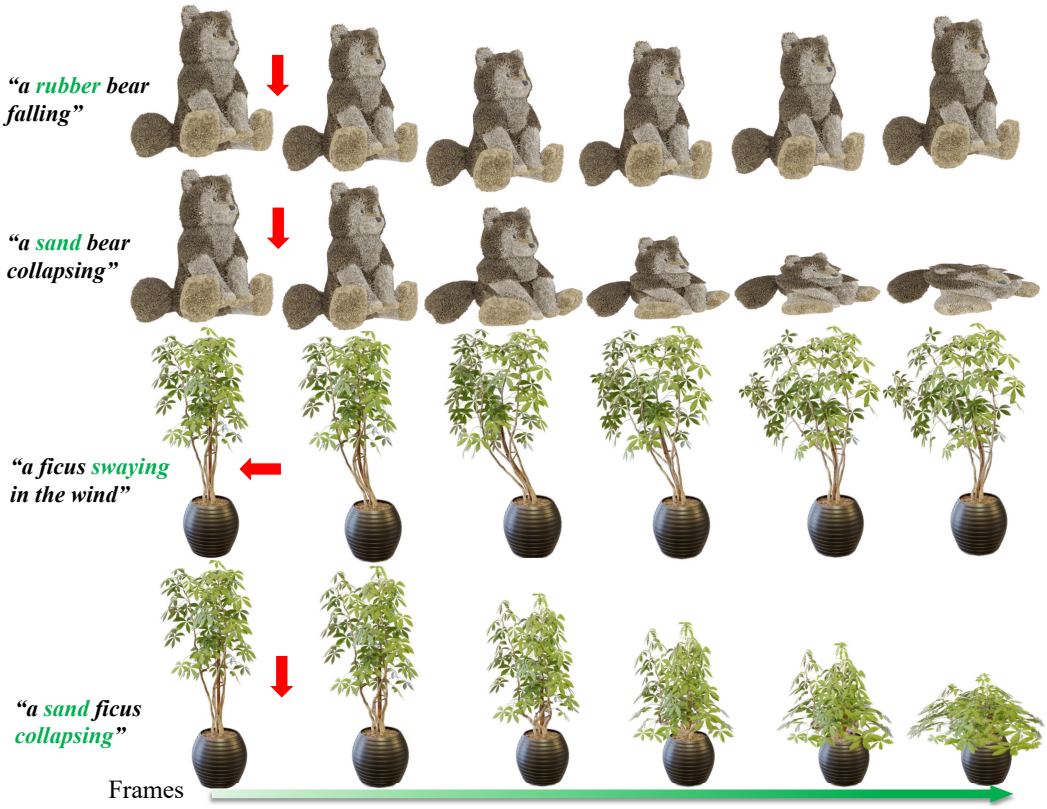


Figure 13: We further demonstrate the capability to simulate different material properties of Fast-PhysGS. For instance, in the bear scene, our method produces bouncy behavior for a rubber material, as well as collapse effects for sand or liquid materials. Meanwhile, ficus achieves soft-body swinging and collapse.

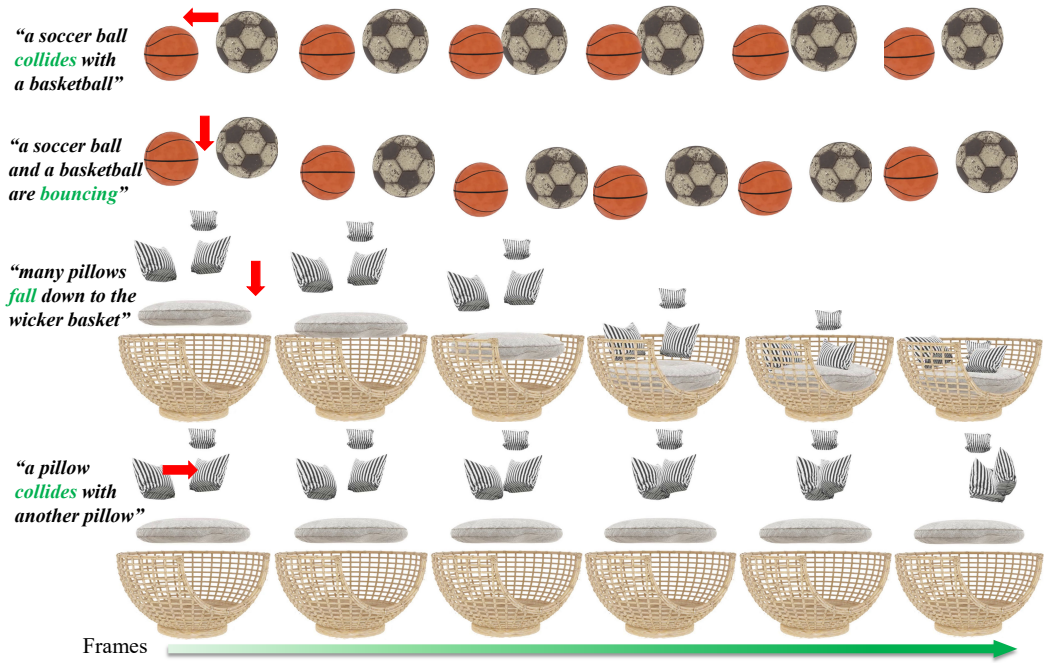


Figure 14: We further demonstrate the ability to simulate diverse motion priors. For example, it can handle the free fall and multi-collisions of objects, such as balls and pillows.



Figure 15: We further demonstrate the ability to simulate diverse motion priors and material properties of different scenes.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [N/A]

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]